JavaScript - Anexo - Uso de **this** en contexto

Una método de un objeto es una función, en el ejemplo la función **getInfo()**:

```
let alumno = {
    nombre: 'Carlos',
    apellidos: 'Pérez Ortiz',
    edad: 19,
};

alumno.getInfo = function() {
    return 'El alumno ' + this.nombre + ' ' + this.apellidos + ' tiene ' + this.edad +
'años'
}

console.log(alumno.getInfo()); //El alumno Carlos Pérez Ortiz tiene 19años
```

También se puede incluir la declaración del método en la declaración del objeto:

```
let alumno = {
    nombre: 'Carlos',
    apellidos: 'Pérez Ortiz',
    edad: 19,
    getInfo: function(){
        return `El alumno ${this.nombre} ${this.apellidos} tiene ${this.edad} años`;
    }
};

console.log(alumno.getInfo()); //El alumno Carlos Pérez Ortiz tiene 19 años
```

OJO: deberíamos poder ponerlo con sintaxis *arrow function*, pero no funciona.

```
let alumno = {
    nombre: 'Carlos',
    apellidos: 'Pérez Ortiz',
    edad: 19,
    getInfo: ()=> `El alumno ${this.nombre} ${this.apellidos} tiene ${this.edad} años`
};

console.log(alumno.getInfo()); //El alumno undefined undefined tiene undefined años
```

Si, es por algo que se llama el **alcance léxico**. Es por el `this` : con la función normal, `this` se refiere al objeto sobre el que se está invocando el método (hasta ahí bien), pero con la **función flecha** `this` se refiere a otra cosa.

Con la **función flecha** `this` se comporta de forma diferente manteniendo el valor que tenía `this` fuera de la **función flecha**.

Sí, `this` en JavaScript estuvo super mal diseñado y depende de la forma en que se invoque la función.

Para hacer ese método declara la función como una función normal.

En general, `this` se refiere al objeto actual.

Pero si no hay un objeto "actual", `this` depende del ambiente: en el navegador se refiere al objeto global `window`, en Node.js es `undefined`.

Así en este caso (navegador) podríamos poner window.innerHeight o this.innerheight indistintamente. En ninguno de los dos casos conseguiremos el objetivo. Veamos:

```
let alumno = {
    nombre: 'Carlos',
    apellidos: 'Pérez Ortiz',
    edad: 19,
    getInfo: ()=> `El alumno ${window.innerHeight} ${this.innerHeight} tiene
${this.edad} años`
};

console.log(alumno.getInfo()); //El alumno 758 758 tiene undefined años
```

En una función normal el valor de `this` puede cambiar dependiendo de cómo se llame la función. Supongamos que tenemos el siguiente método `hello` que utiliza `this` dentro de un objeto:

```
const person = {
  name: "Pedro",
  hello: function hello() {
    console.log(this.name)
  }
}
```

Si invocamos el método `hello` sobre `person` el resultado es el esperado:

```
persona.hello() // "Pedro"
```

Pero uno puede cambiar el `this` en `hello` de varias formas al invocarlo, una de ellas es con el método `call` que recibe como primer argumento lo que uno quiere que sea `this`:

```
persona.hello.call({ name: "Maria" }) // "Maria"
```

En este ejemplo estoy cambiando el `this` para que sea un objeto que tiene una llave `name` con valor "María".

Con las **funciones flecha** no hay forma de cambiar el `this`, que siempre se refiere al `this` que existía cuando se evalúa la función. Esto es más predecible, pero tiene la desventaja que no se puede utilizar en algunos casos, como en los métodos de objetos o funciones constructoras.

Nota: Al valor del `this` también se le conoce como **el contexto**.